A programming technique is presented which permits switching from central to stand-by computer in case of failure. Switchover is accomplished automatically and without loss of data or interruption in service.

The technique is applicable to a large class of commercial real-time systems which must function in an uninterrupted manner.

During the normal periods when both computers are operable, the programming system permits the second computer to be utilized independently for other data processing.

Recovery for computer switchover in a real-time system

by Harry Nagler

In real-time systems which demand uninterrupted servicing of incoming messages, it is customary to employ duplexed central computers, so that one may take the place of the other when needed. Questions arise concerning:

1. The economic justification of the second computer.
2. Keeping the operating personnel constantly alert and prepared for sudden switchover.
3. Loss of information and reduction in service when such switchover occurs.

It is highly desirable to keep the second computer free for independent work while in stand-by status, to make switchover fully automatic, and to maintain full continuity of system performance across the switchover period. A procedure is proposed to achieve these objectives.

This paper is concerned only with the sudden failure of the central computer with attendant complete loss of the contents of its core memory. It is assumed that at the time of failure all other devices of the real-time complex, and in particular the stand-by computer, continue in working order.

system description

The real-time system for which this recovery procedure is designed is first described functionally. Essentially, it is a file maintenance system responsive to messages which may come at any time and from any one of many different places. A message will cause file references, sometimes also file updating, and finally the sending of a response to the originating location within stated time limits. In order to simplify the discussion of the programming techniques used, it will be assumed that no message may be originated at any place where a response to a previous message is still outstanding; but in a later paragraph this restriction is lifted.

The equipment configuration is illustrated in Figure 1. Since the identity of the interchangeable central computers is largely immaterial, they are designated simply as "7000's." They are connected by a signal transmission line. Communication between the 7000 and the remote terminals at which messages originate is coordinated by a message exchange. Since it is assumed to have capability equivalent to that of, say, the IBM® 7750 Programmed Transmission Control, the exchange will be denoted as the "7750." The 7750 need not be duplexed, though if it is, the subsequent program description must be modified. The files may likewise be duplexed or not; if they are not, certain items of information must still be recorded in duplicate. The Tele-Processing® equipment is shown at the top of the diagram, while the equipment shown at the bottom is used by the stand-by 7000 for batch-type processing. Switches permit either 7000 but not both of them together, to be connected to either equipment.

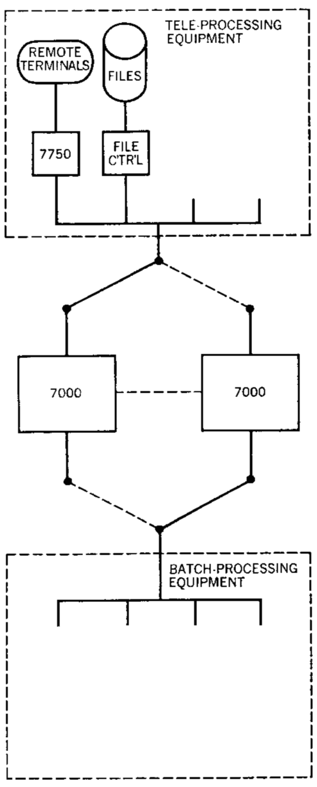


Figure 1 System configuration

control program in stand-by computer

We begin by outlining the conditions to be met by the control program in the stand-by computer. When both running and stand-by 7000's are in working order, they will be in communication at regular intervals. This may take the form of a signal sent by the running to the stand-by 7000 each time the interval timer causes a trap.When such a signal fails to be received in the stand-by 7000 during a certain interval of time, it immediately prepares for takeover by discarding any processing it may be engaged in, and reading in appropriate parts of the control program needed for the recovery procedure. It is assumed that work in progress in the stand-by 7000 is protected by conventional checkpoint and restart procedures. If failure to receive a signal is repeated, the stand-by 7000 will take charge by connecting itself to all input-output devices, thereby disconnecting the presumably failing 7000, and issuing emergency orders to the 7750.

scheduling file references

We must now turn to special conventions about control program action in the 7750 and 7000. A typical message processing path is shown in Figure 2. As each message entering the 7750 from a remote terminal is logged in it will receive a serial number, N. This serial number, together with an origin code, T, will be entered in a 7750 table, referred to as the "7750 message table," from which it will not be purged until the response to this message has left the 7750 again on its way to the terminal. Thus the table reflects at all times the messages which have been received in the central computing complex, but not yet processed to completion. Similarly, when a message enters the 7000, which it does prefixed with its serial number, the serial number will be placed into a table in the 7000, which reflects all messages at some stage of processing within the 7000. This table will be called the "7000 message table." Not until all file updating action caused by this message has been successfully completed and the response successfully transmitted to the 7750, will the corresponding entry be purged from the 7000 message table.

Message serial numbers enter into both ordinary file processing and the recovery procedure. Each record, or field, which may be changed as a result of processing a message contains, besides the field to be updated or consulted, room for the serial number of the message causing the particular file entry to be written. When a message bearing serial number N causes updating of the file, the serial number N^*, associated with the message which caused the last previous updating of the field, will be found recorded alongside that field, and the updating action will include the replacement of N^* by N. However, the entire updating action is dependent on whether N^* is still in the 7000 message table or not; if it is still there, the message with serial number N^* has not yet been processed out of the 7000, and the updating action requested by the message with serial number N must be deferred. What

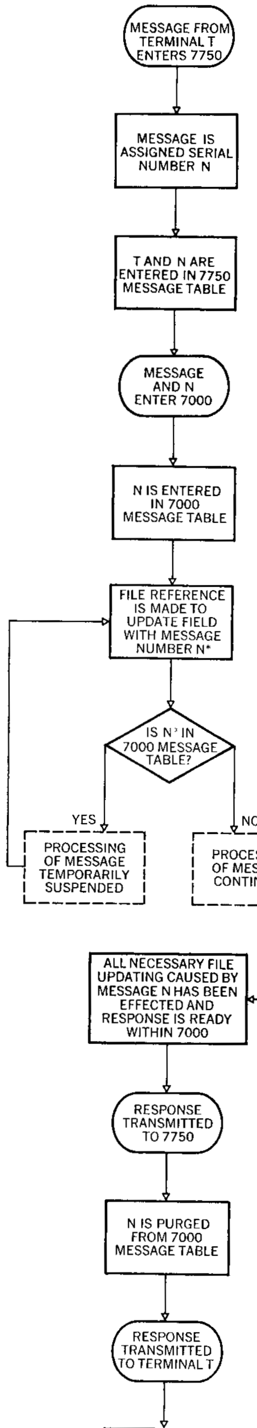
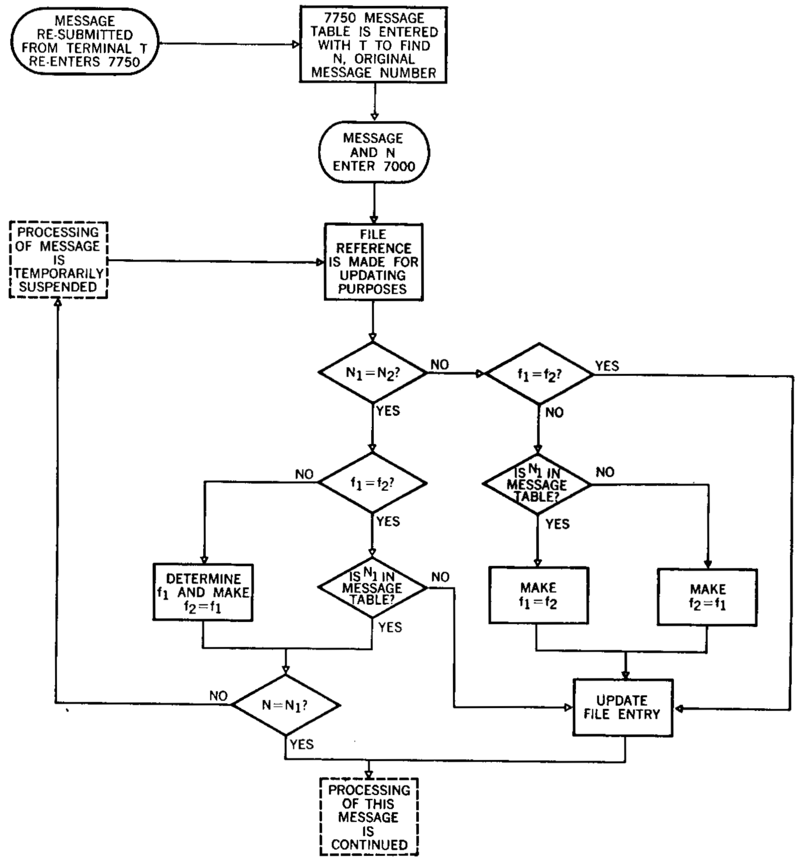

MESSAGE FROM TERMINAL T ENTERS 7750
MESSAGE IS ASSIGNED SERIAL NUMBER N
T AND N ARE ENTERED IN 7750 MESSAGE TABLE
MESSAGE AND N ENTER 7000
N IS ENTERED IN 7000 MESSAGE TABLE
FILE REFERENCE IS MADE TO UPDATE FIELD WITH MESSAGE NUMBER N*
IS N* IN 7000 MESSAGE TABLE?
YES
NO
PROCESSING OF MESSAGE TEMPORARILY SUSPENDED
PROCESSING OF MESSAGE CONTINUED
ALL NECESSARY FILE UPDATING CAUSED BY MESSAGE N HAS BEEN EFFECTED AND RESPONSE IS READY WITHIN 7000
RESPONSE TRANSMITTED TO 7750
N IS PURGED FROM 7000 MESSAGE TABLE
RESPONSE TRANSMITTED TO TERMINAL T
T AND N ARE PURGED FROM 7750 MESSAGE TABLE

Figure 2 Typical path of message processing

Figure 3 Processing of re-submitted messages during recovery


MESSAGE RE-SUBMITTED FROM TERMINAL T RE-ENTERS 7750
7750 MESSAGE TABLE IS ENTERED WITH T TO FIND N, ORIGINAL MESSAGE NUMBER
MESSAGE AND N ENTER 7000
PROCESSING OF MESSAGE IS TEMPORARILY SUSPENDED
FILE REFERENCE IS MADE FOR UPDATING PURPOSES
N1=N2?
NO
YES
f1=f2?
YES
NO
f1=f2?
NO
YES
IS N1 IN MESSAGE TABLE?
NO
YES
DETERMINE f1 AND MAKE f2=f1
IS N1 IN MESSAGE TABLE?
NO
YES
MAKE f1=f2
MAKE f2=f1
NO
N=N1?
YES
UPDATE FILE ENTRY
PROCESSING OF THIS MESSAGE IS CONTINUED

happens is that the current operational program, which is seeking to update the file on behalf of the message with serial number N, will be suspended for the time being and pushed back into the queue of operational programs maintained by the control program. In other words, the file updating called for by that message may not take place until the message which had caused the last previous file updating has been fully processed in the 7000. Of course, if N^* is no longer in the 7000 message table, file updating in response to message N may proceed without delay. This rule makes certain that if a field may be updated by several messages, which can be in the 7000 at the same time, the serial number associated with the field identifies the one and only message which can have caused the updating. This assurance is of use in the recovery procedure.

effects on programming

Under this rule for file updating, an operational program may be suspended not only when it waits for file records to be brought into the 7000 memory, but also when a particular record has been updated too recently. The action of examining the serial number part of a record and deciding whether to proceed with the operational program or not should be standardized so as to be amenable to description by a macro. Better still, it may be possible to incorporate it in the 7000 control program.

switchover procedure

Having described the design features which are to be used for recovery after unscheduled switchover, we next outline the action taken by the control programs in the 7750 and 7000 by which such recovery is effected. When the stand-by computer takes over it will instruct the 7750 to inhibit the acceptance of any further messages from the remote terminals, but will permit it to continue sending completed responses. After all these have been sent, the 7750 message table will be read into the 7000, and the 7000 message table reconstructed from its entries. The two message tables, in the 7750 and 7000, now reflect all messages for which responses are outstanding; and any of these which are no longer recoverable from 7750 memory must be requested again from the associated terminals. It is at this point that use is made of the assumption that no terminal will send a message until response to the last message has been received.

treatment of re-submitted messages

As re-submitted messages come in, they are identified by origin and given the serial number of their originals, which are found by entering the 7750 message table with origin T as the argument. The messages are then transmitted to the 7000, together with any messages recoverable from within 7750 memory. As these messages cause file references in the 7000, the message number associated with the particular file record is examined, and action taken accordingly (Figure 3). If the record bears a serial number not in the 7000 message table, the record may be referenced or updated in the ordinary way. If the serial number is in the 7000 message table, action depends on whether the message causing the updating bears an identical serial number or not. If the serial numbers agree, the file updating in respect to this message had already been effected during the processing of this message's original; only the response message had never reached the 7750. Hence the message is processed in the ordinary way, except that no record is updated if its serial number agrees with that of the referencing message. If, on the other hand, the serial numbers differ, the last updating of this file record was caused by a message whose processing was curtailed by the switchover, and which may or may not yet have been re-submitted; in either event, the normal rule is followed, and the message is queued behind other pending messages which await processing. Because of this procedure, delays in re-submission of messages may delay the processing of other messages more than ordinarily; but this inconvenience is part of the price paid for recovery.

duplicate file entries

During the preceding discussion of the role of message serial numbers during the recovery period, it was tacitly assumed that a file reference would always bring to light a valid file entry. However, since 7000 failure may interfere with the act of writing such a file entry, further safeguards are necessary. If relevant file entries are kept in duplicate versions, written at non-overlapping times, at most one of the two versions can be affected by 7000 failure, and therefore at least one of them will be valid. Let N denote the message serial number, f the field, and let subscripts 1 and 2 refer to the two versions of a file entry. Furthermore, in each file entry, let N be written before f. Then the 7000 writes file entries in the order: N_1, f_1, N_2, f_2. Four cases may be distinguished:

Case 1: $N_1 = N_2$, $f_1 = f_2$. The two versions agree, and the ordinary rules for updating apply, based on whether N_1 is in the 7000 message table or not.

Case 2: $N_1 \neq N_2$, $f_1 = f_2$. File writing was interrupted after beginning to write N_1 but before writing f_1. N_1, if its writing was completed, will be in the 7000 message table. In any event N_2 will not be there. Therefore $f_1 = f_2$ may be taken as the un-updated field, which may now be updated without further delay.

Case 3: $N_1 \neq N_2$, $f_1 \neq f_2$. File writing was interrupted some time after beginning to write f_1 but before the writing of N_2 was complete. N_1 will be in the 7000 message table, and f_2 is the valid field. Therefore, as in Case 2, f_2 may be taken as the un-updated field, which may now be updated without further delay.

Case 4: $N_1 = N_2$, $f_1 \neq f_2$. File writing was interrupted some time after writing N_2 but before the writing of f_2 was complete. f_1 is the valid, updated field. The information contained in the file entries themselves and the 7000 message table does not suffice to determine which of the two discrepant versions of the field is f_1.

format of file entries

The determination of f_1 in the case $N_1 = N_2$, $f_1 \neq f_2$, which will round out the description of the file recovery procedure, involves either the imposition of further constraints upon the system, or the recording of additional information in the file entries. Let us note here that the procedure does not explicitly demand the duplexing of files, so long as N_1, f_1, N_2, and f_2 are written in any convenient place in that order and at non-overlapping times. Thus, they might all be written on the same track of a file, so that f_1 could be determined simply by its position in the track. However, in most real-time systems the files are duplexed for reasons other than protection against possible 7000 failure, and it is not normally practicable to write duplicate file entries in an order determined solely by the two physical file units involved. Thus a knowledge of the physical file unit from which a given field is read is not enough to establish whether it, or its duplicate counterpart, was written first when the field was last updated. Four methods are outlined to settle the matter, all of which affect either the nature or the format of file entries.

Method 1. This applies only if the updated form of a field does not depend on its previous forms; so that updating amounts to replacement, not modification, of the previous contents of the field. This will often be the case with non-numeric information. Let N be the serial number of the message which causes a file reference during the recovery period. If $N = N_1$, the field f_1 may be taken from the updating message, and the discrepancy resolved. If $N \neq N_1$, the message cannot be acted upon further at this time, and must be queued behind other messages. In other words, this message cannot be processed until the message with serial number $N = N_1$ has been re-submitted and processed.

Method 2. The format of file entries provides for an indicator, say i, which is advanced each time the file entry is updated. N, f, and i are written in that order. In its simplest form, i would be a digit which cycles through the three values 0, 1, and 2. Under the assumptions of Case 4 above, i_1 and i_2 are both intact, but different. Reference to their values will establish which of the two versions of the field, f_1 or f_2, is the updated one.

Method 3. This method is based on the requirement that the writing of duplicate versions must not overlap in time. This entails that the control program always knows which version will be written first, and that therefore it can record this priority in the file entry itself. Let j be an indicator which, in its simplest form, will assume the values 0 and 1 for the first and second version respectively. Let j, N, and f be written in that order. Under the assumptions of Case 4 above, j_1 and j_2 are both intact, but different. As in the second method above, reference to their values will establish whether f_1 or f_2 is the updated version of the field.

Method 4. The format of file entries provides for the double recording of the message serial number, such that N, f, and N are written in that order. If, in either file entry, the first and second values of N agree, this entry was not interrupted by 7000 failure and so is valid.

Method 1 requires no additional space in file entries, but is of limited application. Methods 2 and 3 require at most one extra character per file entry, but demand special action on the part of operational and control programs respectively. Method 4 demands the most extra space, but is easiest to program. In a given application a mixture of methods, not including the third, is theoretically possible, but not recommended in practice, since the uniformity of the recovery procedure would be lost.

inquiry without file updating

The preceding discussion has dealt with file references which result in file updating, that is, in changes in the information content of the file. In most real-time systems however, there is a notable volume of inquiries concerned only with eliciting information about the current state of the file, without paying heed to the possibility that subsequent file updating action may render the information out of date, in some cases almost immediately. A large class of inquiries may be processed with fewer constraints than messages involving file updating. Among them are all those which require the extraction of only a single file entry. However, inquiries involving several file references should be treated as though file updating were involved if the correct response depends on the internal consistency of the several references. If file updating were proceeding concurrently, internal consistency of the references might be affected. During normal running of the 7000, the former simpler type of inquiry may be processed without concern for any message serial numbers. During the recovery period, it is only necessary to establish the valid version of the field f, by following the rules stated in previous paragraphs under which, when $N_1 = N_2$, the updated version of f is to be supplied in response to the inquiry; otherwise the un-updated version is to be supplied.

return to normal operation

During the recovery period, the program should keep a check on the response of remote terminals to the request for re-submission of messages; but no way is seen of avoiding erroneous inventories if the request is not complied with. When all messages have been re-submitted, acceptance of further messages from remote terminals may be resumed, and the control programs in both the 7750 and 7000 revert to their normal mode of operation, except for the temporary unavailability of the stand-by 7000.

extensions to more general messages

Where messages may be initiated at remote terminals before a response to previous messages has been delivered, it no longer suffices to use an internal system of serial numbering. The recovery procedure must be able to demand from a given terminal the re-submission, not just of the last message originated, but of any one of several messages on which action had not yet been completed. It is therefore necessary to identify all messages externally, either by requiring the originator to affix an identification, or by letting the 7750 report back to him the assigned serial number immediately after this has been created. With these provisions the recovery procedure works as before.

The scheme outlined above will deal with all messages originating outside the computer complex; however, it will be useful to provide also for the recovery of time-initiated messages, that is, those which are internally generated at predetermined times. Let us consider the intervals between successive signals sent by the running 7000 to its stand-by counterpart. At the beginning of any interval during which time-initiated messages are to be generated, the 7000 will allocate serial numbers to them and record these numbers in the 7750. If time-initiated file references do not interact with file references caused by external messages, it will suffice to allocate numbers from a separate series. As time-initiated outgoing messages leave the 7750, the corresponding entries are purged from the 7750 table for time-initiated messages, in the same way as ordinary ones. In case of switchover, the 7000 will read the 7750 table back into memory, just as it would for ordinary messages, and proceed in the same fashion. However, since there is no way of requesting the re-submission of time-initiated messages, the 7000 must be able to determine what action to take. If this cannot be done from inspecting the serial numbers and the time interval alone, some action code should be appended to the serial numbers in the 7750 message table from which the required action could be determined. Thereafter, updating action is subject to the same rules and constraints as for ordinary messages.

affected areas of systems design

It will be observed that the areas of systems design affected by the present procedure include: action of personnel manning the remote terminals, control program in the 7750, control program in the 7000, record format in files, and record processing by operational programs.

It is therefore seen that precautions against information loss in the case of unscheduled switchover are not confined merely to the control program in the central computer, or to the provision of duplexed equipment, but touch all phases of systems design. This is an important reminder that assuring the reliability of a real-time system is an integral part of overall systems design, and cannot be relegated to an isolated design function.

ACKNOWLEDGMENT

This paper is the outgrowth of work done while the author attended the Advanced Systems Course at the IBM Systems Research Institute in the summer of 1962. Particular thanks are due to J. C. McPherson, director of the institute, for stimulating and constructive comments. In addition, R. V. Head, D. L. Mordy, and W. Selden (all members of IBM) have also contributed many helpful suggestions.